The Virtual Classroom Simulation: Pre-Service Teacher Training With ABMS

Simon Skrådal¹, Sivakumar Alagumalai¹, Michael J. Lawson²
¹Adelaide University, South Australia, School of Education
²Flinders University, South Australia, School of Education

Abstract. The Virtual Classroom Simulation (VCS) spans the fields of Education, Psychology and Computer Sciences. This trans-disciplinary project aims to construct an interactive training tool for pre-service teachers, using concepts from Agent Based Modelling and Simulation (ABMS). In brief, the VCS allows the user (the pre-service teacher) to interact with “simulated” students in a delimited isolated classroom context. By combining aspects of teacher performance, task characteristics and the classroom environment, we aim to monitor how this might influence individual student state and behaviour, as well as how it may correspond to teaching in the real world. Benefits to the user include a more thorough understanding of the pedagogy of teaching and a potential change in their current mental models of teaching. As the tool is likely to trigger discussion amongst users, it also aids in the preparation of pre-service teaching practice and education. The validity of a simulation rests and falls on its underlying theoretical precision and methodology. The system to be simulated must be broken down (deconstructed) into logical fragments of fine granularity that can be reassembled (reconstructed) in the simulation. In this reconstruction stage, one must analyse the problem to be solved and determine what the critical elements of the system are, and filter out those that are not fundamental to the problem solving. This paper examines the VCS from the developer’s perspective, whose task is to translate the educational theory into a mathematical model for the programmer. The process of defining and identifying agents, filtering their restricted set of performance attributes, and how these are influenced by the surrounding context/environment will be discussed. As the VCS is involved with the modelling and simulation of human behaviour and heterogeneous decision making, we will also address, if not stress, the importance of agent delimitation. The paper proposes a conceptual architecture for modelling and simulation of human interactions.

1. INTRODUCTION

1.1 Knowledge for teaching

A number of research studies in education highlight what attributes effective teachers bring to the classroom. Anderson [1] argues that, in order to successfully fulfill their many roles, teachers often resort to personal theories, mental models, schemas and conceptions of education when establishing a good learning environment. In pre-service teachers, such mental models and schemas would mainly be fostered from their years of experience as learners. Both Anderson [1] and Barnes [2] suggest that there is a large knowledge base available to support beginning teachers. Shulman [16] created a classification system of seven types of teacher knowledge that continues to be influential. Shulman’s categories include content knowledge; general pedagogical knowledge; curriculum knowledge; pedagogical content knowledge; knowledge of learners and their characteristics; knowledge of educational contexts; and knowledge of educational ends, purposes and values, and their philosophical and historical grounds. With the addition of Grossman’s [8] ‘knowledge of learning’ category, Shulman’s set of categories provides a sound basis for representation of the range of teacher knowledge. However, as Barnes [2] indicates, teacher education programs too often tend to focus on what pre-service teachers should know in terms of their content knowledge, rather than what they think they already know in terms of their pedagogical content knowledge, knowledge about how to present content to their students [16]. By failing to deal with these preconceptions appropriately, which is what Barnes [2] suggests is happening at large, the teacher education curriculum may also fall short.

Arguably the most significant component of the curriculum that addresses the pedagogical content knowledge is the practice teaching placement, which gives pre-service teachers valuable first-hand teaching experience as well as an opportunity to advance their pedagogical content knowledge. Some research has shown that the practice teaching component is the only part of teacher training that has a considerable influence on pre-service teachers’ existing conceptions about teaching [14]. Similarly, Stones [19] argues that practice teaching is commonly considered to be the single most important component of teacher training.

While contemplating on the future of practice teaching, Stones [19] recommended that it would be even more valuable to couple the teaching placement with an emphasis on self-reflection and discussion between pre-service teachers. Stones [19] argued that expanding this component would assist in developing a better understanding of the body of pedagogical theory and, consequently, increase their pedagogical skills. In order to foster this, Stones [19] indicated that the time devoted to practice teaching could in fact be reduced to allow more time for informed and effective reflective activities. Barnes [2] supports the argument that practical experience in the classroom alone will not
suffice in fostering pre-service teachers’ much needed expertise, nor in challenging their potential misconceptions about teaching. Thus, one could argue that even with a sufficient number of practice teaching placements, there is still a need for extra reflective components in the teacher education program. Hence, coupling the current difficulty of locating available teaching placements with Stones’ [19] and Barnes’ [2] notions of shortcomings in teacher training, it seems appropriate to suggest a rethink of alternate solutions. One area that should be given significant attention is the use of computer technology and simulation models.

Gurvitch’s [9] dissertation on a Computer Mediated Simulation (CMS), designed to enhance task modification decisions among pre-service physical education teachers, shares some of our reasoning and motivation in introducing computer technology in teacher training. However, Gurvitch’s [9] scope is limited to basic sport performance skills, which involve pre-service teachers watching videos and evaluating these in an online sequential questionnaire-style survey whilst receiving feedback on their answers.

Similarly, another interesting classroom simulation described in Kervin et.al [12], aims to engage pre-service kindergarten teachers in deep thinking. In this instance, the approach is a web-based walk-through concentrating on three kids and their learning outcomes, which is affected by the choices made by the user.

Although the VCS share some features of the studies described above, our work parts with the others on many levels. Gurvitch’s [9] study is more akin to an instant-feedback questionnaire than a simulation. The work of Kervin et.al [12] has more in common with the VCS as it addresses desired pedagogical outcomes similar to ours. Their simulation also involves decision making steps that have implications for subsequent decisions. On the other hand, their conceptual design and web-based implementation imply a more restricted and less dynamic simulation than what we aim to develop with the VCS.

1.2 The Virtual Classroom Simulation

This project proposes to further the established conceptualisation and development of an interactive simulation system for training pre-service teachers [18]. A trans-disciplinary effort is underway to develop a Virtual Classroom Simulation (VCS) which introduces a new dimension to pre-service teacher training through a fusion of elements of education theories and Agent Based Modelling and Simulation (ABMS) that promise a new, safe and innovative way for pre-service teachers to interact with virtual students and observe likely trends in student behaviour. By identifying and deconstructing important entities in a teacher, student, teaching and learning environment, we intend to develop an Agent Based Social Simulation (ABSS) [6] that essentially reconstructs and models a classroom teaching scenario. Instructional design lays the basis for extending a sound HCI to a realistic simulation [4]. An appropriate instructional design for system dynamics-based learning needs to incorporate perspectives from learning and learning theories, i.e. designed with clear and consistent elaborate sequences (simple -> complex, breadth -> depth). The system will present to the pre-service teacher a real time interactive simulation where the user can employ a range of methods of teaching with a group of virtual students whilst receiving continuous feedback. In analogical terms, the application will be to the teacher what a flight simulator is to the pilot. We envisage the VCS as a tool that can be used to stimulate challenges to the pre-service teachers’ preconceptions, as well as advancing their mental models and schema of best practice. In turn, reflection on the simulation exercise and its generated output will promote discussion and further enhance pedagogical skills and teaching confidence.

As indicated above, a large amount of the educational theory (e.g. expert identification and deconstruction of entities and attributes) has already been explored and established, we are now at the stage where software development and module implementation may commence. The remainder of this paper will accentuate the design philosophy in many facets of the VCS, ranging from simulation models and agents, user interface and functionality and of course the simulation backend.

As we are still in the early stages of the implementation phase, explicit details on interaction algorithms developed thus far have not been disclosed. We are also in the process of establishing exactly what student/task/teacher attributes to emphasise, as it is not possible to incorporate all of these in the initial rounds of the development cycle. We hope that a fully functional prototype (ready for final stages of user testing) will be complete by the end of 2008.

2. AGENT BASED COMPUTING

Many papers deliver definitions and interpretations of the term ‘Agent’, but exactly what constitutes an agent is ambiguous and highly dependent on the context of implementation (see for example [11], [5], [13], [7] and [20]). To avoid any confusion due to this inherent ambiguity, it is therefore important to clearly define what comprises a VCS Agent:

In the VCS, an agent is a software implementation of a real-world human model, the student, which operates with its own set of beliefs, desires, intentions and emotions (BDIE). The agent model is based on a specific reconstruction of an individual (i.e. a student) derived from an expert deconstruction of what is considered to be essential attributes in a classroom learning context.

In other words, the VCS Agent is a generic software module, which models a student with strictly confined properties and characteristics that is able to mimic real life behaviour. The module is generic in that it can be replicated to any number of individual agents, each of
which can differ in their attribute states, and, hence, display independent behaviour. The two main reasons why it is desirable to keep the agent properties confined relates to a) relevance (i.e. to the context in which the simulation takes place) and b) manageability (i.e. the complexity of the system).

Factors that might influence a student’s behaviour and performance in the classroom are countless. Thus, it would not be possible, nor feasible, to incorporate all of these in the VCS. This is why we stress the importance of finding high priority attributes relevant in our given context (i.e. the classroom) and work with these. Shannon [17] sums it up nicely by drawing comparisons to Pareto’s law of the vital few and trivial many and the importance of including only those aspects relevant to the study. Similarly, Batten [3] argues that it is not always necessary to incorporate all scales of human awareness simultaneously, as they may not be essential to certain social simulations. In our case, the VCS Agent is thus a restricted, though highly specific, representation of a real-world student, designed to express likely behaviour and state changes throughout the course of the simulation.

Simulating individual human behaviour is indeed difficult, and can, at best, only be implemented with educated estimations and approximations. While Batten’s [3] discussion and example of a self-referential system (i.e. the Bar Problem) consider only a single set of beliefs, desires and intentions, VCS agents have potentially several BDIE’s that influence each other. Moreover, unlike the mentioned self-referential system, the VCS complexity is elevated further by its interactive nature. We can use a VCS-related example to illustrate this. One simple attribute, ‘wait-time’, denotes the time it takes from the teacher asking a question to the point when the teacher either prompts a student to answer or makes another comment, perhaps asks another question. What process takes place in real students before they decide to make a contribution? Mood, confidence, question difficulty, topic knowledge, time of day, motivation, other classmates, position in the classroom, attention level, personality, teacher’s expectations, how the student relates to the teacher, etc. are all possible contributors to this process, and they all could have an influence on each other. For example, one student might know the answer well, but refrains from answering due to factors such as shyness, or maybe because the question is so simple that it appears boring and de-motivating. Another student might not know the solution at all, but still responds to every question. It is important for a teacher to observe this behaviour, as the time needed for individual students to generate a possible response varies. If a teacher selects a student too soon, some might miss out, whilst a longer wait time might affect other student attributes (motivation and attention, for example). The teacher might also fall into a pattern where the same student(s) are asked frequently, because they are quick on the mark. So how can we model and implement this seemingly trivial behaviour in an agent, and how can we estimate not only whether or not a student will engage, but also how long the wait time will be? It is evident in both the student and teacher examples that interactions between and within entities are non-deterministic. The simulation, with various mathematical models and algorithms, provides the basis for examining agent-based computing.

The example above raises several implications for our agent, simulation and interface architectures.

### 3. VCS ARCHITECTURES

#### 3.1 VCS Agent Architecture

The ‘wait-time’ attribute demonstrated how a specific behaviour does not at all act as an independent operand; rather it depends and reflects on several attribute traits in the behavioural space of the student, as well as surrounding elements such as ‘question’, ‘teacher’ and ‘classroom climate’. The abovementioned importance of agent delimitation is therefore key in agent modelling and implementation. Consequently, we need to focus on abstraction; i.e. to keep the agents simple by hand-picking a small number of behavioural and task attributes of relatively low complexity and assess the behavioural output of these in isolation. More complexity can then be added to the established platform and assessed continuously.

Prototype implementation will incorporate limited communication (message passing) between the user (teacher) and the students, but not between the agents themselves. The simulation framework will also facilitate a more extensive communication interface for the agents to aid in logging of events and changes in attribute properties.

#### 3.2 VCS Simulation Architecture

Agents plug in to the VCS simulation architecture, which primarily provides standard features such as time-management, message passing and logging. Since the simulation involves human-in-the-loop interactivity, the framework must also supply services for the graphical user interface (GUI).

#### 3.2.1 Simulation Model

Helsgaun [10] lists three simulation model categories:

1. **Continuous**: the state varies continuously with time.
2. **Discrete**: the state changes only at discrete instances of time (event times).
3. **Combined continuous and discrete**: the state changes instantaneously at event times; in between consecutive event times the system state may vary continuously.

VCS agent states might change continuously (e.g. ‘wait-time’ duration causes student attributes to change over time) and discretely (e.g. teacher actions, such as...
‘ask a question’, happen sporadically). Thus, our
proposed simulation model is combined continuous and
discrete, allowing flexibility needed for the variation of
agent states.

3.2.2 Time Management
To make the simulation more realistic, the VCS
predominantly runs in real-time; i.e. 1 wall clock
second maps to 1 simulation second. For certain events,
however, the simulation time will leap ahead of the wall
clock time. Typically, a teacher action that is time
consuming in real life, such as ‘ask a question’, will add
\( n \) seconds to the simulation time whilst resuming
instantly in real time. A user is therefore compelled to
be conscious of time-management issues in the same
way as one would be in a real teaching scenario. In turn,
this has an impact on the simulation model, as we need
to carefully design the simulation clock and consider
how to best couple system state and event-driven
changes.

3.3 VCS Logging Architecture
Several levels of logging must be considered and the
immense quantity of information has to be categorised
and organised appropriately. Aside from the
developer’s run-time logs used for de-bugging, analysis
and validation, the end user should be able to retrieve a
complete and user-friendly representation of the
simulation run. One should be able to examine attribute
details for each agent, when they changed and why. To
achieve this, agent details and events should be coupled
on a timeline, making it easier for the user to examine
consequence \( \leftrightarrow \) event correlations.

3.4 VCS Graphical User Interface
From the user’s perspective, the GUI is perhaps the
most important aspect of the program. This is the
console from which s/he can observe and interact with
the simulation execution (i.e. generating events) as well
as retrieving and examining simulation logs/results.
Aside from obvious execution tasks, the GUI should
also facilitate real-time display of selected agent
attribute states, generation of new students and tasks
and program configuration. Unlike many other types of
software, the VCS targets a highly specific group of
users. Consequently, the GUI must be tailored for this
group according to their preferences and the needs of
their lecturers. Following research on relevant teaching
styles, common/repetitive tasks must be intuitively
placed and easy to recognise, suggesting a number of
prototypes be created and trialled.

Shneiderman [15] highlights the importance of the three
pillars of user-interface architecture for a successful
system. An optimal user-interface design needs to
include ‘usability lab & iterative testing’, ‘user-interface
management system’ and ‘appropriate guidelines and
documents’. Figure 1 highlights Shneiderman’s
conceptions.

From a developer’s perspective, the GUI must be
tightly integrated with the agent, simulation and logging
architectures. How the GUI itself and communication
with the different architectures is implemented is crucial
with regards to performance, as the simulation backend
should be given the majority of system resources.

An early prototype of the VCS GUI, as suggested by
Figure 2, presents the user with a visual display of the
students in the class. Each student has 6 different facial
expressions to indicate their emotional states (happy,
content, normal, tired, sad and upset), along with other
indicators to provide estimated levels of other states
(for example knowledge and motivation). Values are in
estimates in order to make the simulation more realistic
(i.e. a teacher in a real classroom does not have
exhaustive attribute data about their students either).
Detailed logs and charts of student progression,
behaviour and moods will be made available only after
the simulation is complete.
4. TECHNOLOGIES

Considering our architectures’ requirements, potential technologies for implementing the VCS should be carefully assessed.

4.1 Programming Language

After much research, the desired language for implementing the VCS core is Java. VCS in Java means portability, which is an important requirement. Performance concerns with Java have also become less of an issue in later versions, meaning that the VCS should be able to execute without concerns. A large collection of simulation and agent frameworks (open source and commercial) of various complexity and features already exist, and some of these are potentially suitable for the project.

As Java comes with an extensive library, it facilitates for an easier integration with other technologies, such as database connectivity.

5. EVALUATION AND VALIDATION

Three major stages of evaluation and validation (E&V) are apparent in the development of our simulation:

1) The E&V of model input (i.e. identifying appropriate agents, tasks and attributes and how these interrelate)

2) The E&V of the simulation output (i.e. the correctness of simulated results when compared to likely real-world behaviour)

3) The E&V of the simulation’s impact on the users’ schema and pedagogical knowledge

Provided that the simulation algorithms are accurate, the linear correlation between input and output correctness can be represented as:

![Figure 3: VCS Evaluation and Validation](image)

We anticipate a mutual dependency between input and output data (e.g. garbage-in/garbage-out). The second (blurred) curve in Figure 3 suggests the possible pedagogical impact the simulation might have on a user. This refers to the 3rd stage in E&V, which will take place once a stable version of the VCS is available. Although the proposed curve follows the I/O validity, we hypothesise that even an inaccurate simulation could strengthen the users’ pedagogical understanding if they detect such imprecision(s).

We are in an exiting time in education and in particular pre-service teacher education. We now have evidence-based research, educational models and innovative technologies and techniques to enhance the quality of education. As technologies and simulation models mature in their efficiency and complexities, opportunities for enhancing educational practices improve. The VCS offers such an enhancement to improve the proxis of prospective teachers.

REFERENCES


